Gym-ANM Tool – Implementation Document

1. How to use Gym-ANM provided environment – ANMEasy6-v0

This environment has 6 bus network with

- 1 transformer
- 3 loads
- 2 renewable energy sources
- 1 storage unit
- 1 fossil fuel generator used as slack generator
- The observation() component is the identity function in this environment. This leads to a fully observable environment with ot = st.

GitHub link – how to instantiate ANMEasy6 env - https://github.com/anushaihalapathirana/RL-Gym-ANM-tool/blob/master/anmEasy6-test-env.py

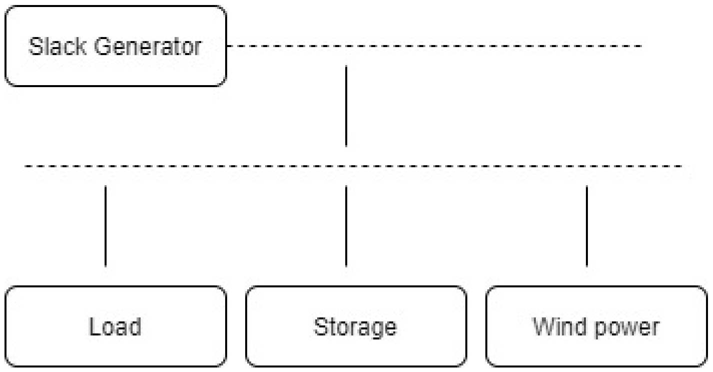
2. Customize ANMEasy6 Environment

We can customize the *Next_vars()* and `observation()` methods

GitHub link - https://github.com/anushaihalapathirana/RL-Gym-ANM-tool/blob/master/customizedEasyenv.py

3. Implement new Environment

I created a simple 3 bus gym-ANM environment with below topology to apply reinforcement learning algorithms.



1.1 How to define new Gym-ANM Environment

Environment – Continuous action and continuous State.

1. Define network using baseMVA, bus, device and branch
2. Implement a sub class of ANMEnv
3. Initialize below variables in __init__ class
    a. Observation space
    b. K – auxiliary variable
    c. Delta_t – time interval
    d. Gamma – discount factor
    e. Lambda – penalty weighting hyperparameter
    f. auxiliary bounds – bounds on auxiliary variable
    g. costs_clipping – reward clipping parameters
    h. seed – random seed
4. Call __init__ method in ANMEnv class (super class)
5. Implement init_state() and next_vars() methods. (observation () method implementation is optional)
6. Init_state() method will initialize the state with random numbers. We can define the state using number of devices, number of storage units and number of generators.

   Number of states in this simple environment,
   State = 2*number of devices + number of DES + number of generators + auxiliary variable
   = 2 * 4 + 1 + 1 + 1 = 11

4. How to use the new Environment

1. Import the environment

   ```
   from python_File_Name import New_Environment_Class_Name
   ```

2. `env = New_Environment_Class_Name()`
3. We can use this env variable like any other openAI gym environment.

GitHub link – how to create and use simple environment -
https://github.com/anushaihalapathirana/RL-Gym-ANM-tool/blob/master/newEnv.py

5. New environment rendering

Implemented rendering to the new simple environment using the rendering code base in Gym-ANM6- easy environment.

This environment is simple 3 bus gym-ANM environment mentioned under Implement new environment topic.

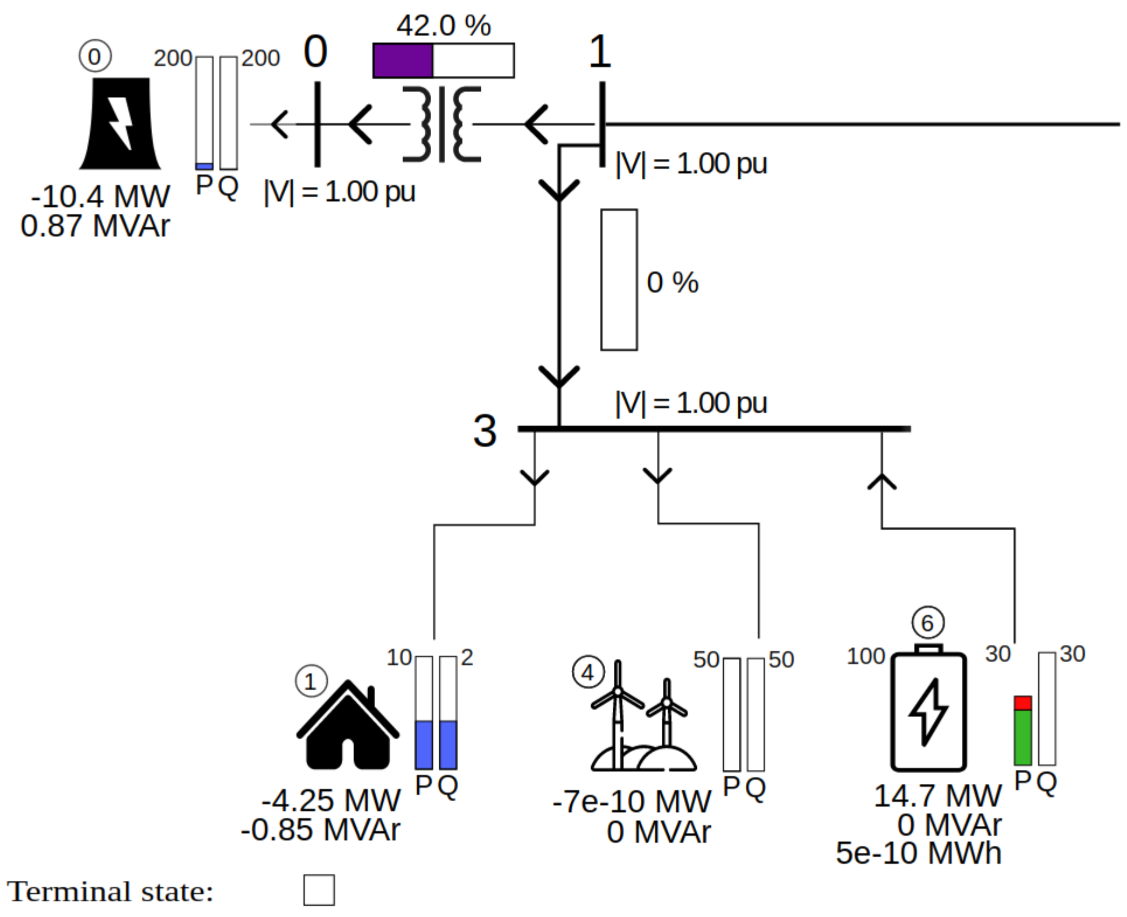


Figure: Snapshot of rendering

<u>Changes</u>

1. Implement new environment, define its network as mentioned in How to define new Gym-ANM Environment

2. Design the network in the network.svg graphic inside /envs/simpleenvironment folder (ex. Use Inkscape tool - https://inkscape.org/ to change svg file.)
3. update the devices, busses, lines and storage units in svgLabels.js (inside /envs/simpleenvironment folder) according to the network.svg (Inkscape provide find functionality to search the id's of objects)
4. Change the Rendering_relative_path in constant.py file if you want
5. Change the title name in index.html file(it is the main rendering file)
6. We can use all the js files inside the js folder as it is. No need to change that if you want to render a simple environment

Github link - How to implement new environment with rendering tool - https://github.com/anushaihalapathirana/Gym-ANM-rendering-tool

Run command – python simpleEnv-instant.py

This will render the environment in browser, similar to the Gym-ANM6-Easy environment.

NOTE - need to implement rendering separately for each new environment

6. AI Techniques

4.1. MPC – Model predictive control

1. Inbuilt to the gym-ANM tool
2. Tool provide 2 MPC agents. Constant policy agent and Perfect policy agent
3. I used constant policy agent to run 10000 steps over my new environmrnt.

Results – 10000 steps cost is 465.65

Code - https://github.com/anushaihalapathirana/RL-Gym-ANM-tool/tree/master/src/MPC

4.2. PPO

Implemented using stable_baselines3 library.
Create a wrapper and use MlpPolicy and train using 10000 time steps
Save the model as ppo_gym_anm_model

Test the model using 10 episodes with 10000 time steps – mean cost is 522.51 and std_cost is 2.418

4.3. SAC

Implemented using stable_baselines3 library.
Modify the action space where all the actions lies in [-1, 1], before creating and train the model
Create SAC model using MlpPolicy, train 10000 time steps and save the model

Test the model using 10 episodes with 10000 time steps – mean cost 526.46 std_cost = 2.639

4.4. A2C

Use 4 parallel environments and create A2C model with MlpPolicy. train 10000 time steps and save the model

Test the model using 10 episodes with 10000 time steps – mean cost 423.04 std_cost – 1.948

A2C, PPO and SAC algorithms can run by running a single main file in the https://github.com/anushaihalapathirana/RL-Gym-ANM-tool/tree/master/src/rl_techniques_baseline3

Note - New model is not created if the model is already saved in the folder. Remove the existing model in the folder if you want to train and save a new model again.

4.5. TD3

Train_td3.py – this file contains the model training related method.

Agent.py – agent class. This class initialize the 2 actor networks and 4 critic networks. Contains choose_action(), remember(), learn(), update_network_parameters(), save_model() and load_models() methods.

Actor.py – The actor neural network to create actor models

Critic.py – critic neural network to create critic models

Buffer.py – contains methods related to replay buffer to store past transitions, which it learns the q-values

Testtd3.py – method to test the trained model

Main.py – main python file. Which will train and test the model.

Train the model using 10 episodes with 10000 time steps and Test the model using 10 episodes with 10000 timesteps – mean cost 499.73 std_cost = 2.485

Note – Can send command line argument to change the environment. To change the environment to the gym-ANM provided simple gym-anm:ANM6Easy-v0, run below command

```
Python src/TD3/main.py DEFAULT
```

Running main.py file without a command line argument will run the simple test environment mentioned above.

GitHub link – TD3 - https://github.com/anushaihalapathirana/RL-Gym-ANM-tool/tree/master/src/TD3

7. Results

Algorithm	**Results (mean cost for 10 episodes with 10000 timesteps)**
MPC	465.65
PPO	522.51
SAC	526.46
A2C	**423.04**
TD3	499.73

References

1. Github – Summer Internship 2021 implementations - https://github.com/anushaihalapathirana/RL-Gym-ANM-tool